Rare sighting of 80% of Bitcoin mining power in one room.
SmartPool

Practical Decentralized Pooled Mining
Mining

Bitcoin:

\[ \text{hash}(\text{BlockHeader}, \text{nonce}) \leq d \]
Ethereum:

\[
\text{hash}(\text{BlockHeader}, \text{nonce}, \text{Dataset}) \leq d
\]
Mining

\[ \text{hash}(\text{BlockHeader}, \text{nonce}, \text{Dataset}) \leq d \]

- Finding a valid nonce is difficult
  - Normal computers take years

- Reward variance
Centralized mining pools – It’s great!

- Join in groups
- Share reward
- Reduce reward variance
Centralized mining pools – It’s great!

- Mining a block:
  \[ \text{hash}(\text{BlockHeader}, \text{nonce}, \text{dataset}) \leq d \]

- Mining a share:
  \[ \text{hash}(\text{BlockHeader}, \text{nonce}, \text{dataset}) \leq D \]
  where \( D \gg d \)

- Every block is a share
- A share is a valid block with probability \( d / D \)
Centralized mining pools – It’s great!

- Pool manager:
  - Check if share is valid
  - Check if share is a block
  - Update miners contribution
  - Pay miners
Centralized mining pools – It’s great!

Highly Centralized

Ethereum’s mining distribution
(https://www.etherchain.org, 7 May 2018)
Centralized mining pools – It’s great!

- Trust
- Censorship
- Pool fee
SmartPool - Motivation

- Decentralize the pool manager
SmartPool - Motivation

- Efficiency
  - Low costs
  - No communication overhead
SmartPool - Motivation

- Security
  - Protect participants from attackers
SmartPool - Motivation

- Fairness
  - Receive rewards in proportion to their contribution
Naive solution
Naive solution

- Smart contract as pool operator
Ethereum Actors

User account:
- Address
- Balance

Smart contract:
- Address
- Balance
- Code
- Private persistent storage
Naive solution

- **Smart Contract:**
  - Check if share is valid
  - Check if share is a block
  - Update miners contribution
  - Pay miners
Naive solution - Problems

- Running a smart contract is not for free
- Fee for submitting a share
- Negative outcome?
Naive solution - Problems

- Transactions are in cleartext
- Attacker can resubmit shares
SmartPool Overview

Miner mines for shares.
SmartPool Overview

Miner submits valid blocks directly to the network.
SmartPool Overview

SmartPool receives the block reward.
SmartPool Overview

Miner wants payment, thus claims shares.
SmartPool Overview

All miners claim their shares.
SmartPool wants a proof from each miner.
SmartPool Overview

Miners submit their Proof.
SmartPool Overview

Each miner receives the reward for his contribution.
How should a miner claim?
SmartPool – Claim

- Submit each share to SmartPool
- Expensive!
SmartPool – Claim

- Claim shares in batches

- Send root of Merkle hash tree
Merkle Hash Tree

- Prove inclusion of S3
- Submit e and d
Merkle Hash Tree - Limitations

- Check uniqueness of a share in one claim?
Merkle Hash Tree - Limitations

- Check uniqueness of a share over multiple claims?
SmartPool - Claim

- Need an ordering
- Each share has an unique counter
  - block timestamp
  - nonce
- Augmented Merkle hash tree
Augmented Merkle Hash Tree

Node = [ min(x), hash(c1 || c2), max(x) ]
Augmented Merkle Hash Tree

Node = [ \text{\texttt{min}}(x), \text{hash}(c_1 \ || \ c_2), \text{max}(x) ]

\begin{itemize}
\item \[ e = [ 1, h(a \ || \ b), 2 ] \]
\item \[ f = [ 3, h(c \ || \ d), 4 ] \]
\item \[ a = [ 1, h(S1), 1 ] \]
\item \[ b = [ 2, h(S2), 2 ] \]
\item \[ c = [ 3, h(S3), 3 ] \]
\item \[ d = [ 4, h(S4), 4 ] \]
\end{itemize}
Augmented Merkle Hash Tree

\[
\text{Node} = [ \min(x), \text{hash}(c1 \ || \ c2), \max(x) ]
\]
Augmented Merkle Hash Tree

- Prove inclusion and ordering of third share
Augmented Merkle Hash Tree

- Prove inclusion and ordering of third share
- Submit e and d

Diagram:

\[
\begin{align*}
    e &= [1, h(a \parallel b), 2] \\
    a &= [1, h(S_1), 1] \\
    b &= [2, h(S_2), 2] \\
    f &= [3, h(c \parallel d), 4] \\
    c &= [3, h(S_3), 3] \\
    d &= [4, h(S_4), 4]
\end{align*}
\]
Augmented Merkle Hash Tree

- Prove inclusion and ordering of third share
Augmented Merkle Hash Tree

- Prove inclusion and ordering of third share
Augmented Merkle Hash Tree

- Prove inclusion and ordering of third share
- Proof fails

\[ a = [1, h(S1), 1] \quad b = [2, h(S2), 2] \quad c = [4, h(S4), 4] \quad d = [3, h(S3), 3] \]

\[ e = [1, h(a || b), 2] \quad f = [3, h(c || d), 4] \]

max counter left child > min counter right child
SmartPool - Claim

Claim = \{ \text{root of augmented Merkle hash tree}, 
\text{number of shares} \}
SmartPool - Verify

- Block
- Network
- Request Proof
- Submit Proof
- Claim
- Reward

SmartPool
SmartPool – Verification

- **Goal:**
  - All shares claimed are valid
  - No share is claimed twice
  - Each share is in at most one claim
SmartPool – Verification

- Verify each share in one claim
- Expensive!
Probabilistic Verification

- Randomly sample one share
Threat Model

- Rational miners
- Deviate arbitrarily from honest protocol to gain more reward
Probabilistic Verification

- Incentive to cheat?

- Penalty scheme for n shares
  - Pay all n shares if invalid share is not detected
  - Pay 0 otherwise
Probabilistic Verification

Alice found 500 valid shares

- She claims 500
  expected payoff = 500

- She claims 1500
  expected payoff = \((1/3) * 1500 + (2/3) * 0 = 500\)

- Alice has not an incentive to cheat
Verifying a Share

- Augmented Merkle hash tree for inclusion and ordering

- Check validity of a share
  - Difficulty
  - Share is constructed correctly
  - Share satisfies PoW
Verifying Ethereum PoW

- Memory hard PoW
- 64 values from 1 GB dataset
- Dataset changes roughly every 4 days
Verifying Ethereum PoW

- Store datasets for the next 10 years in SmartPool
- Future datasets are predictable
- Expensive!
Verifying Ethereum PoW

- SmartPool holds Merkle root of each dataset used in the next 10 years
Verifying Ethereum PoW

- Miner submits
  - 64 values from Dataset
  - Witness (branch of Merkle hash tree)
Verifying Ethereum PoW

- SmartPool compares submitted and stored values

![Diagram of Ethereum PoW verification process involving SmartPool, Merkle root of each dataset, Block, Reward, and Network connections.](Image)
Verifying Ethereum PoW

- Shift solution to the miners

- Miners have to verify the stored values on the contract before joining the pool!
Verifying Ethereum PoW

- **Share Reward**
- **Claim n shares**
- **Prove share i**
- **Share i, Dataset, Witness**
- **Block Reward**
- **EvilPool**
- **Merkle root of each dataset**
- **Eve creates an EvilPool**
- **Manipulate some values**
Each share relying on one manipulated value is invalid. The miner will get payed 0 for the claim.
SmartPool - Results

- Deployed on Ethereum network
- 1 week
- 0.6 % of reward as fees
- 1% - 3% of reward as fees in centralized pools
Discussion

- Decentralization
- Efficiency
- Security
- Fairness

- Bugs?!
- Rely on miner to verify the values stored in the smart contract!
Questions?
## Important fields of block header

<table>
<thead>
<tr>
<th>Field Size (bytes)</th>
<th>Name</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>number</td>
<td>uint</td>
</tr>
<tr>
<td>32</td>
<td>Parent hash</td>
<td>uint</td>
</tr>
<tr>
<td>32</td>
<td>TRIEHASH(TX_list)</td>
<td>uint</td>
</tr>
<tr>
<td>20</td>
<td>Coinbase address</td>
<td>address</td>
</tr>
<tr>
<td>32</td>
<td>state_root</td>
<td>uint</td>
</tr>
<tr>
<td>32</td>
<td>extra_data</td>
<td>char[32]</td>
</tr>
<tr>
<td>8</td>
<td>timestamp</td>
<td>uint</td>
</tr>
<tr>
<td>8</td>
<td>difficulty</td>
<td>uint</td>
</tr>
<tr>
<td>8</td>
<td>nonce</td>
<td>uint</td>
</tr>
</tbody>
</table>